**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Модели данных и системы управления базами данных»

на тему: **«Автоматизированное рабочее место администратора СТО»**

Исполнитель: студентка гр. ИП-31

Процкая М. А.

Руководитель: доцент

Лукьяненко В. О.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2019

**Содержание**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Введение | 3 | | 1 Поставновка задачи | 5 | | 1.1 Требования к проектируемому программному обеспечению | 5 | | [1.2 Обзор](#_Toc468949622) существующих методов и средств автоматизации | 5 | | 1.3 Используемые средства | 8 | | [2 Логическая](#_Toc468949627) модель базы данных | 10 | | 3 Физическая модель базы данных | 13 | | 4 Реализация приложения | 17 | | 5 Тестирование и анализ полученных результатов | 30 | | 5.1 Тестирование пользовательского интерфейса | 30 | | 5.2 Тестирование функционала | 30 | | Заключение | 32 | | Список использованных источников | 23 | | Приложение А | 34 | |

**ВВЕДЕНИЕ**

Задача накопления, обработки и распространения информации стояла перед человечеством на всех этапах его развития. В течение долгого времени основными инструментами для ее решения были мозг, язык и слух человека. Первое кардинальное изменение произошло с приходом письменности, а затем изобретением книгопечатания.

Хранение информации в памяти электронной вычислительной машины (ЭВМ) придает этой информации принципиально новое качество динамичности, т.е. способности к быстрой перестройке и непосредственному ее использованию в решаемых на ЭВМ задачах. Устройства автоматической печати, которыми снабжены современные ЭВМ, позволяют в случае необходимости быстро представить любую выборку из этой информации в форме представления на бумаге.

По мере своего дальнейшего развития административные системы обработки данных переросли в автоматизированные системы управления соответствующими объектами, в которых, как правило, не ограничиваются одной ЭВМ, а в составе двух и более ЭВМ объединяют в вычислительный комплекс.

Интеллектуальные автоматизированные рабочие места (АРМ) в свою очередь также содержат в своем составе ЭВМ, тем или иным способом подсоединенную к центральной автоматизированной системе управления (АСУ). Устройства ввода также должны обеспечивать широкий спектр вводимой информации: текстовой, координатной, факсимильной и т.д. Поэтому АРМ оснащаются при необходимости универсальной или специальной клавиатурой, устройствами ввода координатной информации, различного рода сканерами и т.д.

Развитие рынка автоиндустрии непрерывно развивается и постоянно находится в состоянии трансформации. Трансформация происходит в зависимости от потребностей потребителей данной услуги. Постоянно расширяется сфера услуг и уровень обслуживания.

С каждым годов в Беларуси увеличивается количество автовладельцев, которые предпочитают проходить качественное техническое обслуживание своего автомобиля.

Рост общего количества личных легковых автомобилей по стране за период с 2000 года выглядит следующим образом: их количество увеличилось с 1 385,9 тыс. единиц в 2000-м до 1 737,1 тыс. в 2005-м. В 2010-м насчитывалось уже 2 501,2 тыс., в 2015-м – 2 920,2 тыс., в 2017-м – 2 972,7 тыс. и в конце 2018-го – 3 031 тыс. То есть, с 2000-го количество личных авто выросло более чем вдвое [5].

Согласно правилам проведения государственного технического осмотра механических транспортных средств и прицепов к ним [7], технический осмотр транспортных средств проводится на диагностических станциях Госавтоинспекции (ГАИ), пунктах технического контроля, а также непосредственно на предприятиях. Для проведения технического осмотра транспортных средств на предприятиях, при получении разрешения ГАИ, создаются комиссии.

Поэтому спрос на услуги станций технического обслуживания растет, что говорит о несомненной актуальности тематики курсовой работы.

Станция технического обслуживания (СТО) — организация, предоставляющая услуги населению по плановому техническому обслуживанию, текущему и капитальному ремонтам, устранению неисправностей, установке дополнительного оборудования, восстановительному ремонту автотранспорта.

С точки зрения функционирования или структуры СТО, можно сказать, что она оказывает услуги схожего типа, в частности производит какие-либо манипуляции с автомашиной, приводя ее в рабочее состояние, либо совершенствуя какие-либо ее части.

Все обращающиеся в СТО клиенты, желающие оформить заказ на оказание услуг, должны заполнить форму регистрации, в которой необходимо указать свои ФИО, марку автомобиля, номерной знак и ФИО владельца машины, контактный телефон.

Таким образом, в функционирование СТО входит:

* регистрация клиентов,
* справочная информация о доступных услугах,
* прием заказа на оказание услуг,
* справочная информация о сделанном заказе,
* отчет о проделанных работах и расчет стоимости предоставленных услуг, а также установленных запчастях.

Данные функции на СТО выполняет администратор. Это очень трудоемкая бумажная работа, затрачивающая большое количество времени, поэтому необходимо было разработать АРМ администратора.

Следуя из вышесказанного, можно сказать, что задачами данной работы являются значительное ускорение и повышение качества работы СТО, уменьшение трудоемкости оформления заказа, решение проблем с потерей данных о клиентах, автомобилях, деталях и т.п.

Цель курсовой работы – разработка автоматизированного рабочего места администратора СТО, которым смог бы воспользоваться любой пользователь, имеющий базовые умения для работы с компьютером.

1. **ПОСТАНОВКА ЗАДАЧИ**

**1.1 Требования к проектируемому программному обеспечению**

Согласно заданию к курсовой работе, необходимо разработать приложение «Автоматизированное рабочее место администратора СТО». Оно должно иметь следующий функционал:

* ведение справочников,
* оформление заказов,
* заказ деталей,
* формирование отчетов по запросу.

Данные функции работают на основе следующих справочников:

* Клиент,
* Заказ,
* Сотрудник,
* Должность,
* Ремонтируемая модель,
* Запчасти,
* Услуга,
* Обслуживающие магазины,
* Производитель,
* Авторизация,
* Запчасть для заказа.

Программа должна предоставлять следующие отчеты:

* список всех заказов,
* отчет о предоставляемых услугах,
* формирование остатков запчастей.

1.2 Обзор существующих методов и средств автоматизации

Автоматизированное рабочее место (АРМ) – программно-технический комплекс автоматизированных систем (АС), предназначенный для автоматизации деятельности определенного вида. При разработке АРМ для управления технологическим оборудованием, как правило, используют SCADA-системы.

АРМ объединяет программно-аппаратные средства, обеспечивающие взаимодействие человека с [компьютером](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), предоставляет возможность ввода информации (через [клавиатуру](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0), [компьютерную мышь](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BC%D1%8B%D1%88%D1%8C), [сканер](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BD%D0%B5%D1%80_%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) и пр.) и её вывод на экран [монитора](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80_(%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)), [принтер](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%82%D0%B5%D1%80), [графопостроитель](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%BE%D0%BF%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C), [звуковую карту](https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BA%D0%B0%D1%80%D1%82%D0%B0) — динамики или иные устройства вывода. Как правило, АРМ является частью [АСУ](https://ru.wikipedia.org/wiki/%D0%90%D0%A1%D0%A3).

На сегодняшний день существует огромное количество АРМ. Возможно привести следующие примеры АРМ, в которых значительное место отведено аппаратной реализации:

1) АРМ безопасности "Инспектор +" (рисунок 1.1).

Это комплексное решение для охраны объектов практически любой протяженности и отраслевой принадлежности. Применение уникальных технологий, разработанных специалистами компании, даёт возможность не только осуществлять контроль над охраняемой территорией с помощью подсистем, входящих в "Инспектор + ", но и проводить комплексный анализ ситуации, основываясь на данных от различных модулей. Это становится возможным, поскольку подсистемы представляют собой не отдельно работающие модули, а интегрированные части одной большой программы, которая называется АРМ «Инспектор +».

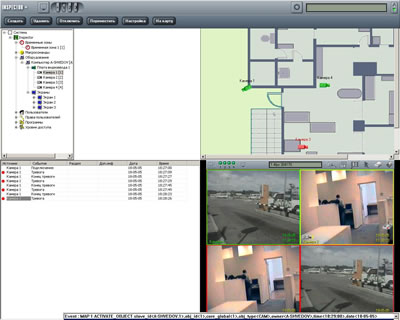


Рисунок 1.1 – АРМ «Инспектор+»

2) АРМ дежурного оператора пункта централизованной охраны НПО «Центр - ПРОТОН» (рисунок 1.2).

Данное автоматизированное рабочее место оператора является функциональным расширением радиосистемы «Радиус» для обеспечения отображения полной карты состояния объектов, ведения базы данных объектов и статистики принятых сообщений.

В состав АРМ входит ПЭВМ (компьютер) типа IBM PC на базе процессора Pentium и программное обеспечение ПО АРМ «Радиус».

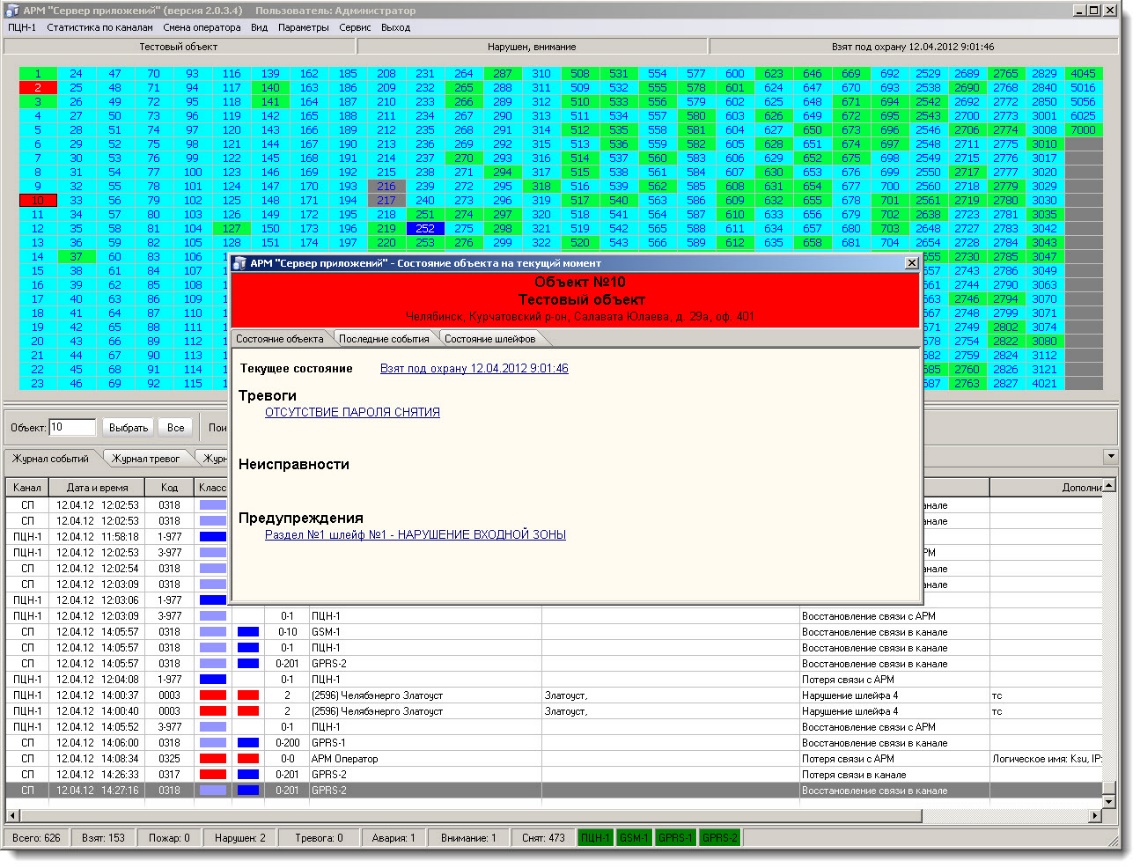


Рисунок 1.2 – АРМ «Центр-ПРОТОН»

3) АРМ для системы среднего и высшего образования.

Группа компаний "Формоза" представляет АРМ для учителя, для библиотекаря и для школьного администратора. Каждое из предлагаемых АРМ — это полностью укомплектованное и готовое к работе учебное оборудование. Внедрение АРМ — это один из шагов к решению такой актуальной задачи, как создание в образовательном учреждении единого информационного пространства.

Учебное оборудование АРМ учителя включает в себя мультимедийный компьютер TEEN на базе процессора Intel(r) Рentium(r) 4 с предустановленным ПО (Windows 2000, Office XP RUS) и с полным набором функций, необходимых для полноценного процесса обучения. В комплект АРМ входит мультимедиа-проектор Hitachi CP-X 275. Его использование позволяет учителю при необходимости переносить компьютерное и видеоизображение на большой экран. Проектор прост в эксплуатации, для работы с ним не требуется никаких специальных знаний и навыков. Также преподаватель может воспользоваться системой "видеокамера - проектор - экран" например, при проведении химического или физического опыта. И экран (размер - 150х150 см, на штативе), и цифровая видеокамера (D-LINK DSC-350), и даже стол для проектора включены в комплект оборудования.

4) АРМ - кладовщика склада готовой продукции ООО «ИнфоТек- Сервис».

Данный АРМ предназначен для:

1. учета передачи продукции со склада в транспортное средство покупателя,
2. оперативного учета движения продукции по складу готовой продукции,
3. автоматизации процесса выписки отгрузочно-сопроводительных документов,
4. проведение инвентаризации продукции на складе готовой продукции,
5. ведения в реальном времени наличия изделий на складе.

На компьютер, установленный на складе, подключается оптический интерфейс OR20 для сброса данных с карандаша MW25. Кладовщики обеспечиваются карандашами MW25. Кладовщик при отгрузке (приемке) продукции карандашом считывает серийный номер холодильника. После окончания загрузки осуществляет сброс информации с карандаша на компьютер с автоматической привязкой номера рабочего места (табельный номер кладовщика), дата и время. Данные сохраняются в компьютере. Таким образом можно гибко изменять количество мест отгрузки и приемки продукции и их расположение.

**1.3 Используемые средства**

Одними из самых популярных языков, используемых для разработки Windows-приложений, являются C# и Java.

C# – это язык программирования общего назначения, который впервые появился в 2000 году в рамках инициативы Microsoft .NET. Он был разработан для общей языковой инфраструктуры (CLI) – открытой спецификации, разработанной Microsoft и стандартизированной ISO и ECMA.

Приложения C# скомпилированы в байт-код, который может запускаться при реализации CLI.

Java, первоначально выпущенный Sun Microsystems в 1995 году, является языком программирования общего назначения, который был разработан с конкретной целью, позволяющей разработчикам написать код единожды и запускать в любом месте.

Java-приложения скомпилированы в байт-код, который может запускаться при реализации виртуальной машины Java (JVM). Подобно CLI, JVM помогает преодолеть разрыв между исходным кодом и 1 и 0, которые понимает компьютер.

C# в основном используется в разработке для платформ Microsoft – .NET Framework CLR и является наиболее широко используемой реализацией CLI. На другом конце спектра Java имеет огромную экосистему с открытым исходным кодом и у него открылось второе дыхание отчасти благодаря тому, что Google

использует JVM для Android [8].

В данной курсовой работе главными частями являются создания приложения под Windows, создание сервера для взаимодействия с базой данных и создание базы данных. Исходя из вышеперечисленного, можно заметить, что язык C# является более подходящим для поставленной задачи.

Синтаксис C# довольно прост в изучении. Все, кто знаком с языками C, C++ или Java с легкостью узнают синтаксис с фигурными скобками, характерный для языка C#. Разработчики, знающие любой из этих языков, как правило, смогут добиться эффективной работы с языком C# за очень короткое время.

Синтаксис C# делает проще то, что было сложно в C++, и обеспечивает мощные возможности, такие как типы значений Nullable, перечисления, делегаты, лямбда-выражения и прямой доступ к памяти, чего нет в Java.

C# поддерживает универсальные методы и типы, обеспечивая более высокий уровень безопасности и производительности, а также итераторы, позволяющие при реализации коллекций классов определять собственное поведение итерации, которое может легко использоваться в клиентском коде.

Выражения LINQ делают строго типизированный запрос очень удобной языковой конструкцией.

Для хранения данных и управления ими используется система управления реляционными базами данных Microsoft SQL Server. Эта система максимально упрощает процесс работы с данными, имеет понятный пользователю интерфейс и представляет большое количество возможностей для разработки.

Работа над проектом велась в среде Visual Studio, так как она является наиболее качественной для разработки на языке C# для операционной системы Windows и предоставляет наиболее быстрые, удобные и многофункциональные средства разработки программного обеспечения.

Разработка проекта велась в среде Visual Studio 2019.

**2 ЛОГИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ**

Для отображения информационной модели для разрабатываемого продукта были введены следующие сущности:

* Клиент,
* Заказ,
* Сотрудник,
* Должность,
* Ремонтируемая модель,
* Запчасти,
* Услуга,
* Поставщик,
* Производитель,
* Авторизация,
* Запчасть для заказа.

Сущность «Клиент» должна хранить список клиентов, которые могут оформлять заказы на СТО. В ней должны присутствовать следующие атрибуты:

* ID,
* ФИО,
* Адрес,
* Номер телефона.

Сущность «Сотрудники» должна хранить информацию о всех сотрудниках. В ней должны присутствовать следующие атрибуты:

* ID,
* ФИО,
* Возраст
* Адрес,
* Номер телефона,
* Серия и номер паспорта,
* Внешний ключ ID Должности.

Сущность «Должности» должна хранить в себе список должностей, занимаемых сотрудниками на СТО. В ней должны быть следующие атрибуты:

* ID,
* Название,
* Зарплата
* Обязанности,
* Требования.

Сущность «Ремонтируемые модели» должна хранить в себе список автомобилей, находящихся в базе данных СТО. В ней должны быть следующие атрибуты:

* ID,
* Модель,
* Производитель,
* Дата выпуска,
* Дата последнего техосмотра.

Сущность «Услуги» должна хранить в себе список услуг, оказываемых на СТО. В ней должны быть следующие атрибуты:

* ID,
* Название,
* Цена.

Сущность «Производители» должна хранить в себе список производителей, поставляющих детали поставщикам. В ней должны быть следующие атрибуты:

* ID,
* Наименование компании,
* Адрес,
* Телефон,
* Внешний ключ ID Поставщики.

Сущность «Поставщики» должна хранить в себе список поставщиков, поставляющих детали на СТО. В ней должны быть следующие атрибуты:

* ID,
* Наименование магазина,
* Адрес,
* Телефон.

Сущность «Заказы» должна хранить в себе список заказов, выполняемых на СТО. В ней должны быть следующие атрибуты:

* ID,
* Внешний ключ ID Услуги,
* Внешний ключ ID Клиента,
* Внешний ключ ID Сотрудника,
* Отметка о выполнении,
* Стоимость,
* Дата оформления заказа,
* Дата возврата,
* Внешний ключ ID Модели.

Сущность «Запчасти» должна хранить в себе список должностей, занимаемых сотрудниками на СТО. В ней должны быть следующие атрибуты:

* ID,
* Наименование,
* Цена,
* Наличие,
* Внешний ключ ID Поставщики.

Сущность «Авторизация» необходима для хранения данных для авторизации, необходимых для входа администратора в систему. В ней должны быть следующие атрибуты:

* Логин,
* Пароль.

Сущность «Запчасть для заказа» является промежуточной для отсутствия связи «многие-ко-многим». В ней должны быть следующие атрибуты:

* Внешний ключ ID Запчасти,
* Внешний ключ ID Заказа.

Логическая модель базы данных представлена на рисунке 2.1.

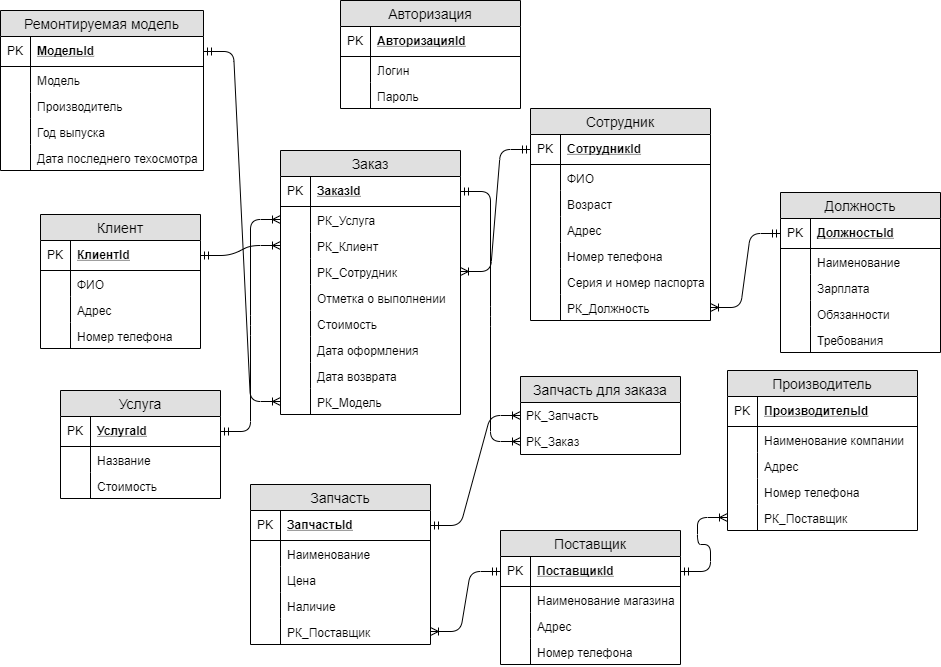


Рисунок 2.1 – Логическая модель базы данных

3 ФИЗИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ

Физическая модель БД определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Построение физической модели базы данных производилось на основе логической модели, представленной в разделе 2. В таблице 3.1 описано соответствие сущностей логической модели и таблиц физической. Подробное описание каждой сущности приведено таблицах 3.2-3.12.

Таблица 3.1 – **Соответствие сущностей логической и физической моделей**

|  |  |
| --- | --- |
| **Сущность** | **Таблица** |
| Клиент | Client |
| Заказ | Orders |
| Сотрудник | Worker |
| Должность | Position |
| Ремонтируемая модель | Model |
| Запчасти | Spares |
| Услуга | Service |
| Поставщик | ServicedShops |
| Производитель | SpareManufacturer |
| Авторизация | Authorization |
| Запчасть для заказа | SpareForModel |

Таблица 3.2 – **Таблица Client**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| FIO | nvarchar(50) |
| Adress | nvarchar(50) |
| PhoneNum | int |

Таблица 3.3 – **Таблица Orders**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| PK\_Service | nvarchar(50) |
| **Имя столбца** | **Тип** |
| PK\_Client | nvarchar(50) |
| PK\_Worker | float |
| ExecutionMark | nvarchar(50) |
| Price | nvarchar(50) |
| OrderDate | nvarchar(50) |
| ReturnDate | nvarchar(50) |
| PK\_Model | int |

Таблица 3.4 – **Таблица Worker**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| FIO | nvarchar(50) |
| Age | nvarchar(50) |
| Adress | nvarchar(50) |
| PhoneNum | nvarchar(50) |
| Passport | nvarchar(50) |
| PK\_Position | nvarchar(50) |

Таблица 3.5 – **Таблица Position**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| PositionName | nvarchar(50) |
| Salary | nvarchar(50) |
| Duties | float |
| Requirements | nvarchar(50) |

Таблица 3.6 – **Таблица Model**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| ModelName | nvarchar(50) |
| Manufacturer | nvarchar(50) |
| YearOfManufacture | float |
| InspectionDate | nvarchar(50) |

Таблица 3.7 – **Таблица Spares**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| SpareName | nvarchar(50) |
| Price | nvarchar(50) |
| Existence | float |
| PK\_Shop | nvarchar(50) |

Таблица 3.8 – **Таблица Service**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| ServiceName | nvarchar(50) |
| Price | nvarchar(50) |

Таблица 3.9 – **Таблица ServicedShops**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| ShopName | nvarchar(50) |
| Adress | nvarchar(50) |
| PhoneNum | float |

Таблица 3.10 – **Таблица SpareManufacturer**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Id | int |
| ManufacturerName | nvarchar(50) |
| Adress | nvarchar(50) |
| PhoneNum | float |
| PK\_Shop | nvarchar(50) |

Таблица 3.11 – **Таблица Authorization**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| Login | int |
| Password | nvarchar(50) |

Таблица 3.12 – **Таблица SpareForOrder**

|  |  |
| --- | --- |
| **Имя столбца** | **Тип** |
| PK\_Spare | int |
| PK\_Order | nvarchar(50) |

Схема физической модели представлена на рисунке 3.1.

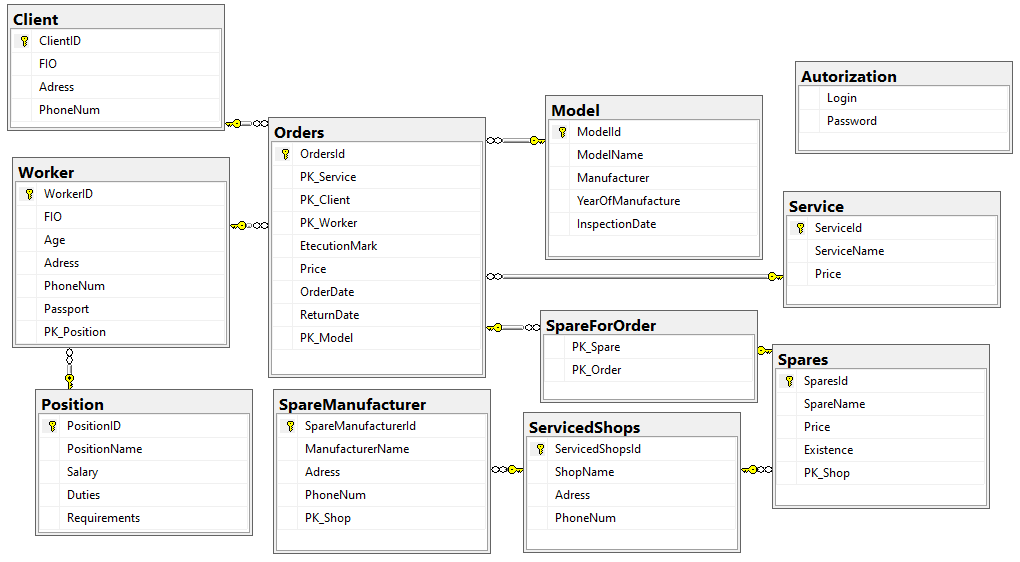


Рисунок 3.1 – Схема базы данных для СТО

Как видно из схем, между таблицами присутствуют связи «один-ко-многим» или «многие-к-одному». При дальнейшем сопровождении приложения и по требованию заказчика, приложение может быть обновлено и улучшено не только добавлением новых таблиц и связей, но и новых функций.

4 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Разработанное приложение имеет следующие окна:

* окно авторизации;
* главное окно;
* окно для добавления ремонтируемой модели;
* окно для добавления нового сотрудника;
* окно для добавления нового клиента;
* окно для обновления информации о ремонтируемой модели;
* окно для обновления информации о клиенте;
* окно для обновления информации о сотруднике;
* отчет о выполненных заказах;
* отчет об осуществляемых услугах;
* окно, позволяющее осуществить заказ запчастей.

Всё приложение построено на технологии Windows Forms. Это обусловлено возможностью уделить больше внимания на функционал приложения, нежели интерфейсу.

В окне авторизации, представленном на рисунке 4.1, предоставляется возможность ввести логин и пароль для дальнейшей работы с программой. При нажатии кнопки «Вход» происходит обработка введенных данных. Если они введены правильно, то пользователь переходит в главное окно (рисунок 4.3). Если логин и пароль неверны, появляется сообщение с предложением повторить попытку (рисунок 4.2).

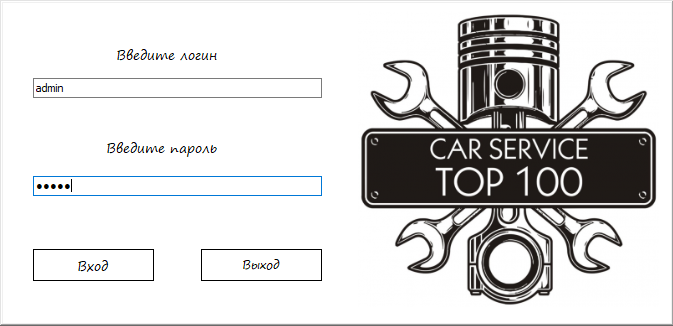


Рисунок 4.1 – Окно авторизации администратора

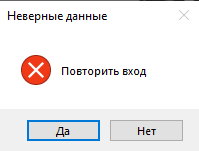


Рисунок 4.2 – Сообщение при вводе неверных данных

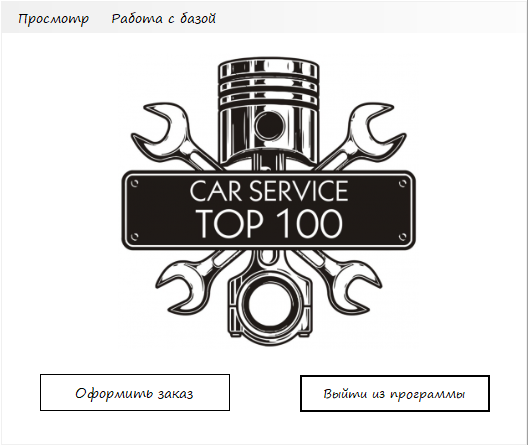


Рисунок 4.3 – Главное окно приложения

Сверху администратор может наблюдать два выпадающих меню (рисунки 4.4 и 4.5): для просмотра данных и работы непосредственно с базой. Внизу расположены две кнопки. При нажатии на кнопку «Оформить заказ» администратор переходит на форму для добавления нового заказа (рисунок 4.7). Если пользователь нажимает на кнопку «Выйти из программы», появляется предупреждение для подтверждения действия (рисунок 4.6).



Рисунок 4.4 – Выпадающее меню для просмотра данных

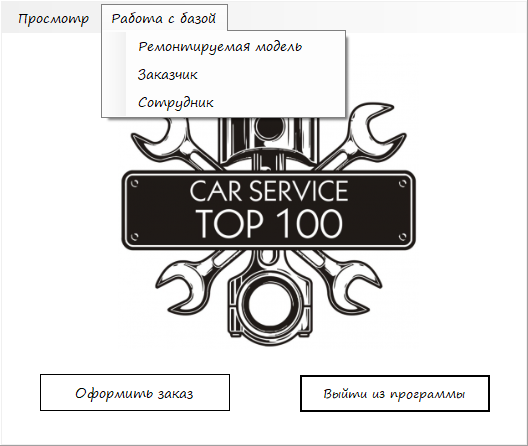


Рисунок 4.5 – Выпадающее меню для работы с базой данных

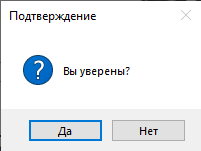


Рисунок 4.6 – Подтверждение действия

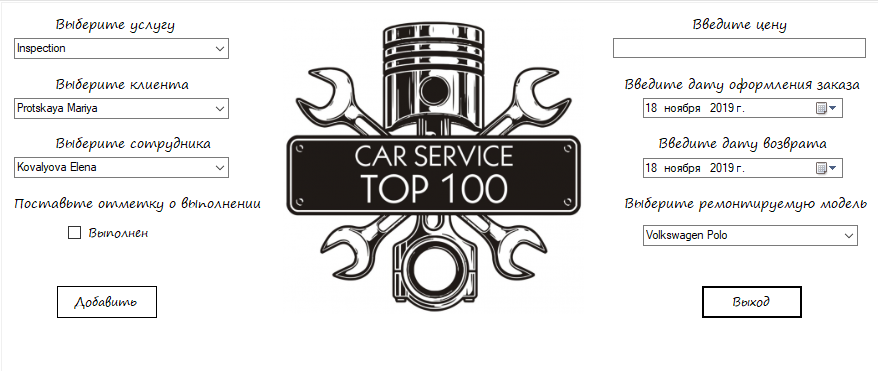


Рисунок 4.7 – Окно добавления нового заказа

Рассмотрим подробнее первое выпадающее меню (рисунок 4.4). При выборе пункта «Просмотреть все заказы», пользователь переходит в окно просмотра списка заказов, оформленных в базе (рисунок 4.8).

При нажатии на кнопку «Отмена», пользователь также увидит предупреждение для подтверждения действия (рисунок 4.6).

При нажатии на кнопку «Да», пользователь вернется на главное окно (рисунок 4.3).

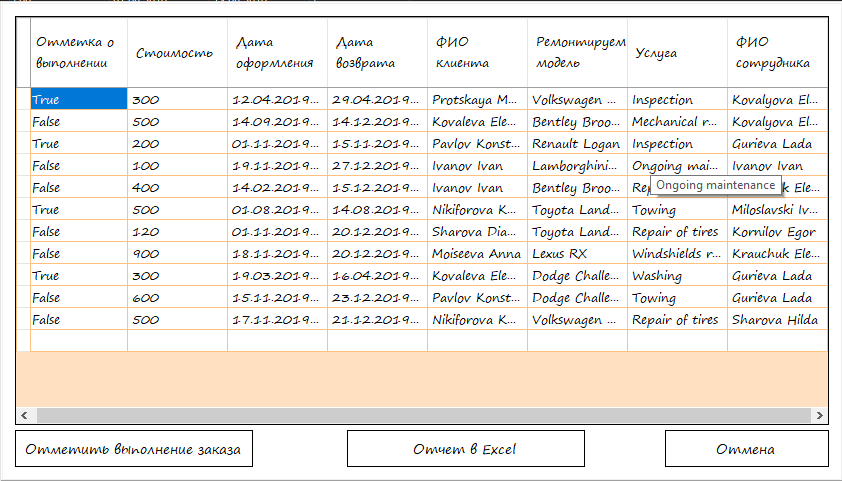


Рисунок 4.8 – Список всех заказов

Окно со списком заказов содержит кнопку «Отчет в Excel», нажав на которую, пользователь получает отчет о всех заказах, выполняемых на СТО, в виде данных, выгруженных в Excel. Результат составления отчета представлен на рисунке 4.9.

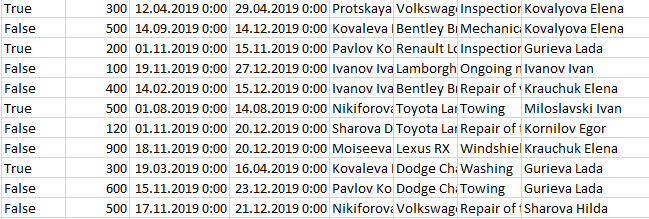


Рисунок 4.9 – Отчет о заказах, отправленный в Excel

При выборе пункта «Просмотреть все услуги» пользователю откроется окно со списком всех осуществляемых на СТО услуг и цен на них (рисунок 4.10). Окно содержит кнопку «Отмена», при нажатии на которую пользователь увидит сообщение о подтверждении действия (рисунок 4.6). При нажатии на кнопку «Да», пользователь вернется на главное окно (рисунок 4.3).

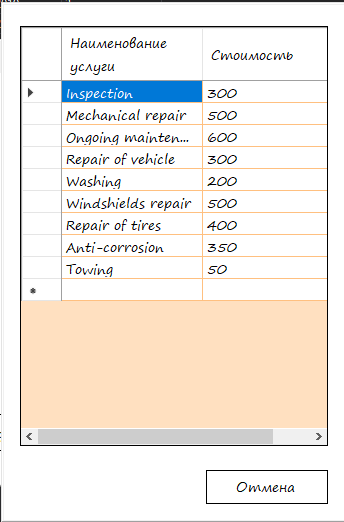


Рисунок 4.10 – Список осуществляемых услуг

При выборе пункта «Проверить наличие запчастей» пользователю откроется окно со списком отсутствующих на СТО запчастей с информацией о поставщиках и производителях (рисунок 4.11).

Окно содержит кнопку «Отмена», при нажатии на которую пользователь увидит сообщение о подтверждении действия (рисунок 4.6).

При нажатии на кнопку «Да», пользователь вернется на главное окно (рисунок 4.3).

Также присутствует кнопка «Заказать», при нажатии на которую осуществляется заказ выбранных запчастей (рисунок 4.12).

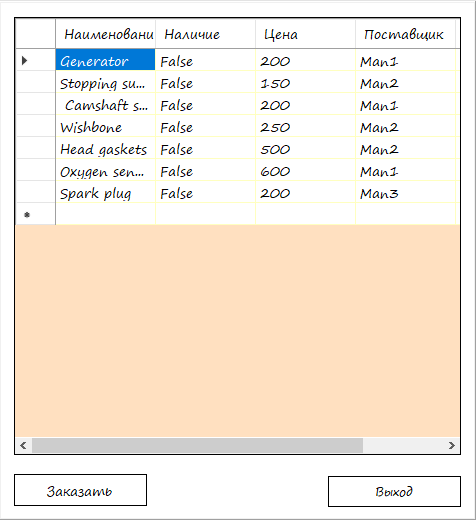


Рисунок 4.11 – Отсутствующие детали

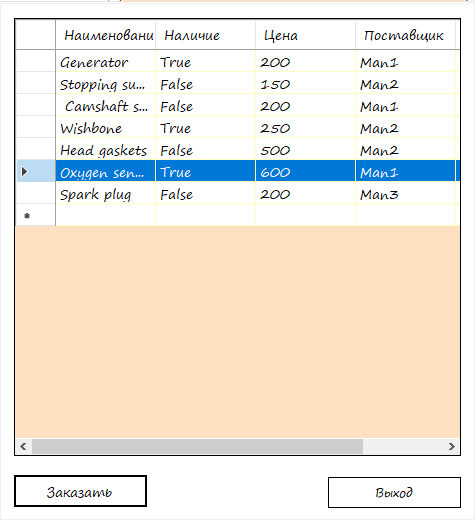


Рисунок 4.12 – Деталь заказана

При выборе пункта меню «Просмотреть выполненные заказы», открывается окно, содержащее список заказов, которые были выполнены ранее на СТО (рисунок 4.13).

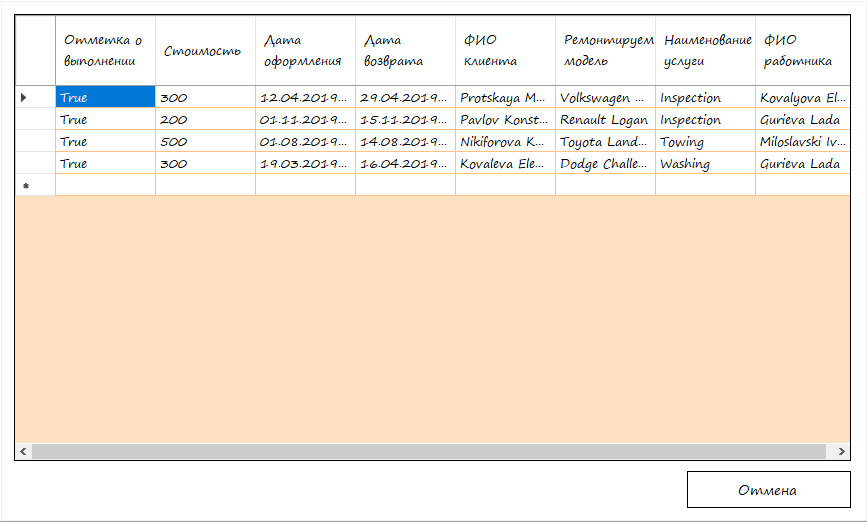


Рисунок 4.13 – Просмотреть выполненные заказы

Рассмотрим второе выпадающее меню (рисунок 4.5). Здесь администратор может выбрать коллекцию для последующей работы с ней.

Первый пункт – работа с ремонтируемыми моделями. Выбрав его, пользователю открывается окно для работы с данной коллекцией (рисунок 4.14).

Слева присутствует список всех ремонтируемых моделей и данные о них. Справа – четыре кнопки.

Нажав кнопку «Добавить», пользователю открывается окно для добавления (рисунок 4.15).

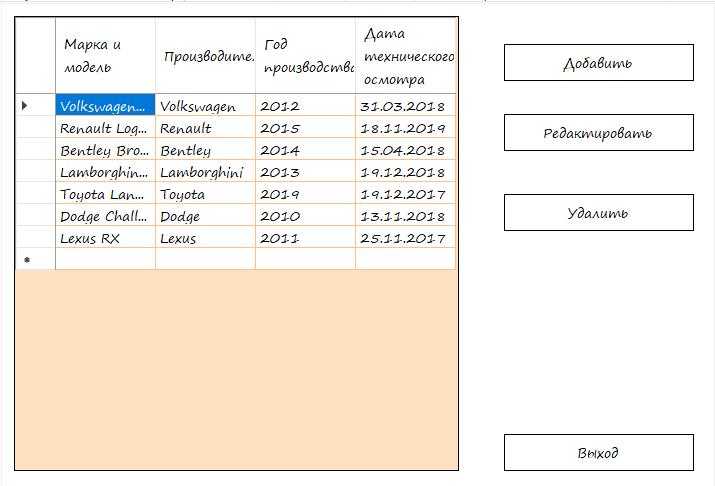


Рисунок 4.14 – Работа с ремонтируемыми моделями



Рисунок 4.15 – Добавление модели

Нажав на кнопку «Редактировать», пользователь переходит в окно редактирования информации (рисунок 4.16), где может обновить информацию о выбранной модели.

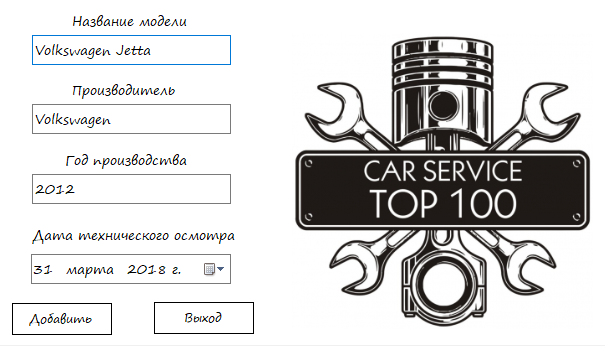


Рисунок 4.16 – Редактирование информации

При нажатии на кнопку «Удалить», пользователь получит подтверждение о выполнении удаления (рисунок 4.6). При ответе «Да», выбранная запись удалится.

Следующий пункт в меню – работа с коллекцией Клиент. Выбрав данный пункт, пользователю открывается окно для работы с клиентами (рисунок 4.17).

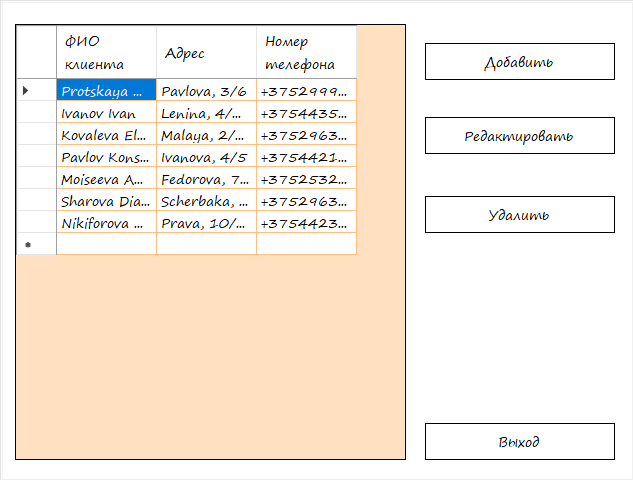


Рисунок 4.17 – Работа с клиентами

Пользователь может добавить (рисунок 4.18), удалить клиента из базы, а также отредактировать информацию (рисунок 4.19) о выбранном клиенте.

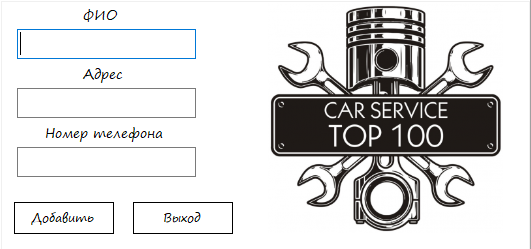


Рисунок 4.18 – Добавление клиента

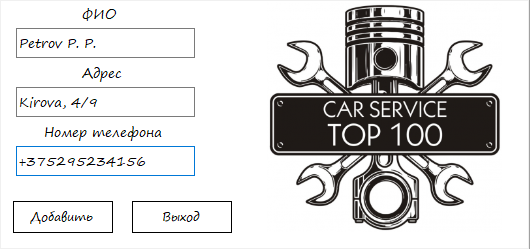


Рисунок 4.19 – Редактирование информации

При нажатии на кнопку «Удалить», пользователь получит подтверждение о выполнении удаления (рисунок 4.6). При ответе «Да», выбранная запись удалится.

Третий пункт в меню – работа с коллекцией Сотрудник. Выбрав данный пункт, пользователю открывается окно для работы с работниками СТО   
(рисунок 4.20).

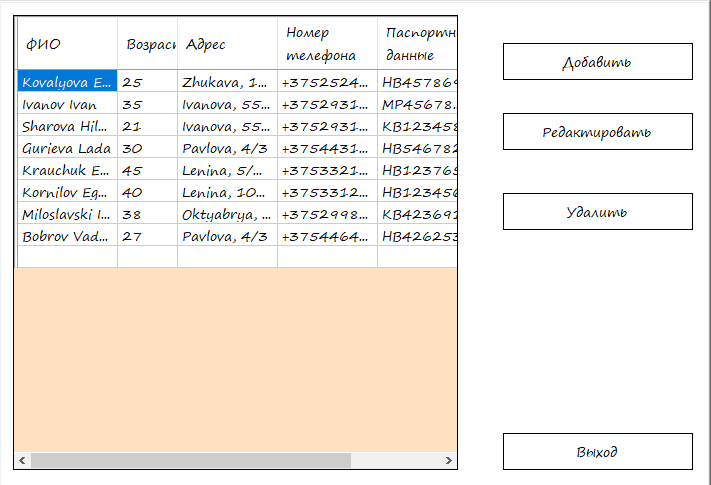


Рисунок 4.20 – Работа с сотрудниками

В данном окне присутствуют кнопки «Добавить», «Редактировать», «Удалить». При нажатии на первую, пользователь видит окно для добавления нового сотрудника (рисунок 4.21).



Рисунок 4.21 – Добавление сотрудника

При нажатии на кнопку «Редактировать», открывается окно для редактирования информации о выбранном сотруднике (рисунок 4.22).



Рисунок 4.22 – Редактирование информации

При нажатии на кнопку «Удалить», пользователь получит подтверждение о выполнении удаления (рисунок 4.6). При ответе «Да», выбранная запись удалится.

**5 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ**

**5.1 Тестирование пользовательского интерфейса**

Весь необходимый функционал приложения был реализован с помощью графических элементов. Все действия в приложении сопровождаются соответствующими уведомлениями. Для каждого уведомления был подобран соответствующий дизайн: шрифт, цвет и содержание. Для приложения были подобраны стили, которые делают приложение более привлекательным, но при этом не мешают читабельности. Приложение имеет наиболее удобный интерфейс. Всё приложение выполнено в одном стиле. В приложении подобраны сочетающиеся друг с другим цветом. Отсутствуют орфографические и пунктуационные ошибки. При наведении на определённые элементы, меняется курсор.

В ходе тестирования приложения были просмотрены основные моменты, на которых программа могла работать некорректно. Результаты тестирования пользовательского интерфейса приведены в таблице 5.1.1.

Таблица 5.1.1 – **Тестирование пользовательского интерфейса**

|  |  |  |
| --- | --- | --- |
| **№** | **Вид проверки** | **Результат** |
| 1 | Реализуется ли функционал приложения при помощи графических элементов | 10 |
| 2 | Реализуется ли размещение всех сообщений об ошибках, уведомлений | 10 |
| 3 | Адаптация под разные размеры экрана | 10 |
| 4 | Отсутствуют ли орфографически, пунктуационные ошибки | 10 |
| 5 | Читабелен ли используемый шрифт | 9 |
| 6 | Изменяется ли вид курсора при наведении на различные элементы | 10 |

**5.2 Тестирование функционала**

Приложение было успешно протестировано на возникновение исключительных ситуаций.

Если пользователь вводит неверные логин и/или пароль при авторизации, он получает уведомление о вводе некорректных данных с предложением повторить попытку (рисунок 5.2.1).

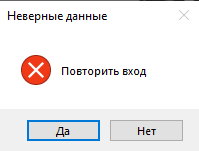


Рисунок 5.2.1 – Сообщение при авторизации

Если пользователь нажимает кнопку «Отмена» или «Выход» (например, рисунки 4.7 – 4.8), он получает сообщение с предложением подтвердить действие, а именно закрыть окно (рисунок 5.2.2). Это сообщение необходимо для того, чтобы предотвратить случайное нажатие и, в следствие, несохранение данных.

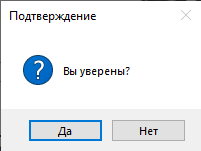


Рисунок 5.2.2 – Сообщение при случайном нажатии

Если пользователь ошибается при вводе данных, он также получает соответствующее сообщение (рисунок 5.2.3).

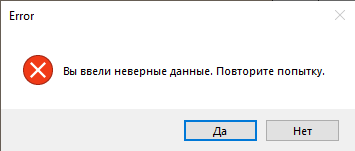


Рисунок 5.2.3 – Сообщение при неверном вводе

**ЗАКЛЮЧЕНИЕ**

В курсовой работе было создано приложение, предназначенное для автоматизации рабочего места администратора СТО. В ходе работы была определена постановка задачи. Поставленная задача была решена при помощи объектно-ориентированного языка C#, с использованием технологии Windows Forms и базы данных MS-SQL Server 2017, с использованием среды Microsoft SQL Server Management Studio 2019.

В ходе разработки проекта, в нем были учтены и обработаны различные ошибки, исключительные ситуации, которые могли бы возникнуть у пользователя во время работы с данным продуктом. Программа обладает небольшим функционалом, однако, несмотря на это, имеющийся функционал является применимым для данной предметной области. Также в ходе подготовки данной программы было выявлено, что аналогов на данный момент практически нет, что повышает её возможный спрос, ведь она упрощает работу как самой СТО, так и повышает уровень комфорта для клиента.

Проект был реализован с возможностью дальнейшего развития.

Поставленные задачи курсовой работы выполнены полностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. – 3-е изд., – СПб.: Питер, 2014. – 816 с.
2. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд., – М.: Вильямс, 2013. – 1312 с.
3. Шилдт, Г. С# 4.0: Полное руководство – M.: Вильямс, 2010. – 1056 с.
4. .NET Framework 4.6 и 4.5. – Электронн. данные. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/w0x726c2(v=vs.110).aspx. – Дата доступа: 19.11.2019.
5. Myfin.by. – Электронн. данные. – Режим доступа: <https://myfin.by/stati/view/13235-skolko-avto-prihoditsya-na-1000-belorusov>. – Дата доступа: 19.11.2019.
6. AutoKatalog. – Электронн. данные. – Режим доступа: <https://autokatalog.by/article/774/>. – Дата доступа: 19.11.2019.
7. Кодексы Беларуси. – Электронн. данные. – Режим доступа: <http://pravo.levonevsky.org/bazaby/org335/basic/text0532.htm>. – Дата доступа: 19.11.2019.
8. Hiring / Development / C# vs. Java. – Электронн. данные. – Режим доступа: <https://www.upwork.com/hiring/development/c-vs-java/>. – Дата доступа: 19.11.2019.

**ПРИЛОЖЕНИЕ А**

AddSpares.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseProject

{

public partial class AddSpares : Form

{

public AddSpares()

{

InitializeComponent();

LoadData();

}

private void CheckSpares\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Spares". При необходимости она может быть перемещена или удалена.

//this.sparesTableAdapter.Fill(this.serviceStationDataSet.Spares);

}

private void LoadData()

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

SqlConnection myConnection = new SqlConnection(connectString);

myConnection.Open();

string query = "SELECT Spares.SpareName, Spares.Existence, Spares.Price, ServicedShops.ShopName, SpareManufacturer.ManufacturerName FROM Spares INNER JOIN ServicedShops ON Spares.PK\_Shop = ServicedShops.ServicedShopsId INNER JOIN SpareManufacturer ON SpareManufacturer.PK\_Shop = ServicedShops.ServicedShopsId WHERE Spares.Existence = 'False'";

SqlCommand command = new SqlCommand(query, myConnection);

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[5]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][3] = reader[4].ToString();

}

reader.Close();

myConnection.Close();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

private void заказатьBtn\_Click(object sender, EventArgs e)

{

int ind = dataGridView1.SelectedCells[0].RowIndex;

dataGridView1[1, ind].Value = true;

//dataGridView1.Rows.RemoveAt(ind);

dataGridView1.Refresh();

}

}

}

AllOrders.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseProject

{

public partial class AllOrders : Form

{

public AllOrders()

{

InitializeComponent();

LoadData();

}

private void AllOrders\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Orders". При необходимости она может быть перемещена или удалена.

//this.ordersTableAdapter.Fill(this.serviceStationDataSet.Orders);

}

private void LoadData()

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

SqlConnection myConnection = new SqlConnection(connectString);

myConnection.Open();

string query = "SELECT Orders.EtecutionMark, Orders.Price, Orders.OrderDate, Orders.ReturnDate, Client.FIO, Model.ModelName, Service.ServiceName, Worker.FIO FROM Orders INNER JOIN Client ON Orders.PK\_Client = Client.ClientID INNER JOIN Model ON Orders.PK\_Model = Model.ModelId INNER JOIN Service ON Orders.PK\_Service = Service.ServiceId INNER JOIN Worker ON Orders.PK\_Worker = Worker.WorkerID";

SqlCommand command = new SqlCommand(query, myConnection);

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[8]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

data[data.Count - 1][6] = reader[6].ToString();

data[data.Count - 1][7] = reader[7].ToString();

}

reader.Close();

myConnection.Close();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

private void button2\_Click(object sender, EventArgs e)

{

copyAlltoClipboard();

Microsoft.Office.Interop.Excel.Application xlexcel;

Microsoft.Office.Interop.Excel.Workbook xlWorkBook;

Microsoft.Office.Interop.Excel.Worksheet xlWorkSheet;

object misValue = System.Reflection.Missing.Value;

xlexcel = new Microsoft.Office.Interop.Excel.Application();

xlexcel.Visible = true;

xlWorkBook = xlexcel.Workbooks.Add(misValue);

xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)xlWorkBook.Worksheets.get\_Item(1);

Microsoft.Office.Interop.Excel.Range CR = (Microsoft.Office.Interop.Excel.Range)xlWorkSheet.Cells[1, 1];

CR.Select();

xlWorkSheet.PasteSpecial(CR, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, true);

}

private void copyAlltoClipboard()

{

dataGridView1.SelectAll();

DataObject dataObj = dataGridView1.GetClipboardContent();

if (dataObj != null)

Clipboard.SetDataObject(dataObj);

}

}

}

AllServices.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class AllServices : Form

{

public AllServices()

{

InitializeComponent();

}

private void AllServices\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Service". При необходимости она может быть перемещена или удалена.

this.serviceTableAdapter.Fill(this.serviceStationDataSet.Service);

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

Authorization.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class Authorization : Form

{

public Authorization()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

private void enterBtn\_Click(object sender, EventArgs e)

{

string login = textBox1.Text;

string password = textBox2.Text;

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

SqlConnection myConnection = new SqlConnection(connectString);

myConnection.Open();

string query = "SELECT Auth.AuthLogin, Auth.AuthPass FROM Auth";

SqlCommand command = new SqlCommand(query, myConnection);

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[8]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

}

reader.Close();

myConnection.Close();

foreach (string[] s in data)

{

if (login == s[0] && password == s[1])

{

//Hide();

MainForm mainForm = new MainForm();

mainForm.ShowDialog();

}

else

{

DialogResult result = MessageBox.Show(

"Повторить вход",

"Неверные данные",

MessageBoxButtons.YesNo,

MessageBoxIcon.Error);

if (result == DialogResult.Yes)

Refresh();

// this.TopMost = false;

break;

}

}

}

}

}

Clients.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class Clients : Form

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

public Clients()

{

InitializeComponent();

}

private void AllClients\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Client". При необходимости она может быть перемещена или удалена.

this.clientTableAdapter.Fill(this.serviceStationDataSet.Client);

}

private void добавитб\_Click(object sender, EventArgs e)

{

string SqlText;

NewClient newClient = new NewClient();

if(newClient.ShowDialog() == DialogResult.OK)

{

// сформировать SQL-строку

SqlText = "INSERT INTO [Client] ([FIO], [Adress],[PhoneNum]) VALUES (";

SqlText = SqlText + "\'" + newClient.textBox1.Text + "\', ";

SqlText = SqlText + "\'" + newClient.textBox3.Text + "\', ";

SqlText = SqlText + "\'" + newClient.textBox4.Text + "\')";

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

}

dataGridView1.Refresh();

}

private void редактировать\_Click(object sender, EventArgs e)

{

int index, n;

string SqlText = "UPDATE [Client] SET ";

string ID\_Source, name, address;

// проверка, есть ли вообще записи в таблице Source

n = dataGridView1.Rows.Count;

if (n == 1) return;

UpdClient f = new UpdClient();

// заполнить форму данными перед открытием

index = dataGridView1.CurrentRow.Index;

ID\_Source = dataGridView1[0, index].Value.ToString();

name = dataGridView1[1, index].Value.ToString();

address = dataGridView1[2, index].Value.ToString();

/\*f.textBox1.Text = name;

f.textBox2.Text = address;

if (f.ShowDialog() == DialogResult.OK)

{

name = f.textBox1.Text;

address = f.textBox2.Text;

SqlText += "Name = \'" + name + "\', Address = '" + address + "\' ";

SqlText += "WHERE [Source].ID\_Source = " + ID\_Source;

MyExecuteNonQuery(SqlText);

FillSource();

}\*/

}

private void удалить\_Click(object sender, EventArgs e)

{

int index, n;

string ID\_Client;

string SqlText = "DELETE FROM [Client] WHERE [Client].ClientID = ";

// проверка, есть ли вообще записи в таблице Source

n = dataGridView1.Rows.Count;

if (n == 1) return;

index = dataGridView1.CurrentRow.Index;

ID\_Client = Convert.ToString(dataGridView1[0, index].Value);

// сформировать SQL-команду

SqlText = SqlText + ID\_Client;

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

dataGridView1.Refresh();

}

this.TopMost = true;

/\*if (f.ShowDialog() == DialogResult.OK) // вывести форму

{

// отобразить таблицу Source

FillSource();

}\*/

}

private void выход\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

public void MyExecuteNonQuery(string SqlText)

{

SqlConnection cn; // экземпляр класса типа SqlConnection

SqlCommand cmd;

// выделение памяти с инициализацией строки соединения с базой данных

cn = new SqlConnection(connectString);

cn.Open(); // открыть источник данных

cmd = cn.CreateCommand(); // задать SQL-команду

cmd.CommandText = SqlText; // задать командную строку

cmd.ExecuteNonQuery(); // выполнить SQL-команду

cn.Close(); // закрыть источник данных

}

}

}

ExecOrders.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class ExecOrders : Form

{

public ExecOrders()

{

InitializeComponent();

LoadData();

}

private void LoadData()

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

SqlConnection myConnection = new SqlConnection(connectString);

myConnection.Open();

string query = "SELECT Orders.EtecutionMark, Orders.Price, Orders.OrderDate, Orders.ReturnDate, Client.FIO, Model.ModelName, Service.ServiceName, Worker.FIO FROM Orders INNER JOIN Client ON Orders.PK\_Client = Client.ClientID INNER JOIN Model ON Orders.PK\_Model = Model.ModelId INNER JOIN Service ON Orders.PK\_Service = Service.ServiceId INNER JOIN Worker ON Orders.PK\_Worker = Worker.WorkerID WHERE Orders.EtecutionMark = 'TRUE'";

SqlCommand command = new SqlCommand(query, myConnection);

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[8]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

data[data.Count - 1][6] = reader[6].ToString();

data[data.Count - 1][7] = reader[7].ToString();

}

reader.Close();

myConnection.Close();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

private void ExecOrders\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Orders". При необходимости она может быть перемещена или удалена.

// this.ordersTableAdapter.Fill(this.serviceStationDataSet.Orders);

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

private void fillByToolStripButton\_Click(object sender, EventArgs e)

{

try

{

this.ordersTableAdapter.FillBy(this.serviceStationDataSet.Orders);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

private void fillByToolStripButton1\_Click(object sender, EventArgs e)

{

try

{

this.ordersTableAdapter.FillBy(this.serviceStationDataSet.Orders);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

}

}

MainForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class MainForm : Form

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

public MainForm()

{

InitializeComponent();

}

private void всеЗаказыToolStripMenuItem\_Click(object sender, EventArgs e)

{

AllOrders allOrders = new AllOrders();

allOrders.ShowDialog();

}

private void найтиЗаказПоФамилииToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void проверитьНаличиеДеталейToolStripMenuItem\_Click(object sender, EventArgs e)

{

AddSpares checkSpares = new AddSpares();

checkSpares.ShowDialog();

}

private void просмотретьВсеУслугиToolStripMenuItem\_Click(object sender, EventArgs e)

{

AllServices allServices = new AllServices();

allServices.ShowDialog();

}

private void запчастьToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

private void NewOrderBtn\_Click(object sender, EventArgs e)

{

NewOrder newOrder = new NewOrder();

//Hide();

string SqlText;

if (newOrder.ShowDialog() == DialogResult.OK)

{

// сформировать SQL-строку

SqlText = "INSERT INTO [Orders] ([PK\_Service], [PK\_Client],[PK\_Worker],[EtecutionMark],[Price],[OrderDate],[ReturnDate],[PK\_Model]) VALUES (";

SqlText = SqlText + "\'" + newOrder.comboBox1.ValueMember + "\', ";

SqlText = SqlText + "\'" + newOrder.comboBox2.ValueMember + "\', ";

SqlText = SqlText + "\'" + newOrder.comboBox3.ValueMember + "\', ";

SqlText = SqlText + "\'" + newOrder.checkBox1.Text + "\', ";

SqlText = SqlText + "\'" + newOrder.textBox1.Text + "\', ";

SqlText = SqlText + "\'" + newOrder.dateTimePicker1.Value + "\', ";

SqlText = SqlText + "\'" + newOrder.dateTimePicker2.Value + "\', ";

SqlText = SqlText + "\'" + newOrder.comboBox4.ValueMember + "\')";

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

}

}

public void MyExecuteNonQuery(string SqlText)

{

SqlConnection cn; // экземпляр класса типа SqlConnection

SqlCommand cmd;

// выделение памяти с инициализацией строки соединения с базой данных

cn = new SqlConnection(connectString);

cn.Open(); // открыть источник данных

cmd = cn.CreateCommand(); // задать SQL-команду

cmd.CommandText = SqlText; // задать командную строку

cmd.ExecuteNonQuery(); // выполнить SQL-команду

cn.Close(); // закрыть источник данных

}

private void МодельToolStripMenuItem\_Click(object sender, EventArgs e)

{

Models models = new Models();

models.ShowDialog();

}

private void заказчикToolStripMenuItem\_Click(object sender, EventArgs e)

{

Clients clients = new Clients();

clients.ShowDialog();

}

private void работникToolStripMenuItem\_Click(object sender, EventArgs e)

{

Workers workers = new Workers();

workers.ShowDialog();

}

private void просмотретьВыполненныеЗаказыToolStripMenuItem\_Click(object sender, EventArgs e)

{

ExecOrders execOrders = new ExecOrders();

execOrders.ShowDialog();

}

}

}

Models.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class Models : Form

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

public Models()

{

InitializeComponent();

}

private void AllModels\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Model". При необходимости она может быть перемещена или удалена.

this.modelTableAdapter.Fill(this.serviceStationDataSet.Model);

}

private void редактировать\_Click(object sender, EventArgs e)

{

int index, n;

string SqlText = "UPDATE [Model] SET ";

string ID\_Source, name, address;

// проверка, есть ли вообще записи в таблице Source

n = dataGridView1.Rows.Count;

if (n == 1) return;

UpdModel f = new UpdModel();

// заполнить форму данными перед открытием

index = dataGridView1.CurrentRow.Index;

ID\_Source = dataGridView1[0, index].Value.ToString();

name = dataGridView1[1, index].Value.ToString();

address = dataGridView1[2, index].Value.ToString();

/\*f.textBox1.Text = name;

f.textBox2.Text = address;

if (f.ShowDialog() == DialogResult.OK)

{

name = f.textBox1.Text;

address = f.textBox2.Text;

SqlText += "Name = \'" + name + "\', Address = '" + address + "\' ";

SqlText += "WHERE [Source].ID\_Source = " + ID\_Source;

MyExecuteNonQuery(SqlText);

FillSource();

}\*/

}

private void выход\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

private void удалить\_Click(object sender, EventArgs e)

{

int index, n;

string ID\_Model;

string SqlText = "DELETE FROM [Model] WHERE [Model].ModelID = ";

// проверка, есть ли вообще записи в таблице Source

n = dataGridView1.Rows.Count;

if (n == 1) return;

index = dataGridView1.CurrentRow.Index;

ID\_Model = Convert.ToString(dataGridView1[0, index].Value);

// сформировать SQL-команду

SqlText = SqlText + ID\_Model;

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

dataGridView1.Refresh();

}

this.TopMost = true;

}

private void добавитб\_Click(object sender, EventArgs e)

{

string SqlText;

NewModel newmodel = new NewModel();

if (newmodel.ShowDialog() == DialogResult.OK)

{

// сформировать SQL-строку

SqlText = "INSERT INTO [Model] ([ModelName], [Manufacturer],[YearManufacture], [InspectionDate]) VALUES (";

SqlText = SqlText + "\'" + newmodel.textBox1.Text + "\', ";

SqlText = SqlText + "\'" + newmodel.textBox2.Text + "\', ";

SqlText = SqlText + "\'" + newmodel.textBox3.Text + "\', ";

SqlText = SqlText + "\'" + newmodel.dateTimePicker1.Value + "\')";

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

}

dataGridView1.Refresh();

}

public void MyExecuteNonQuery(string SqlText)

{

SqlConnection cn; // экземпляр класса типа SqlConnection

SqlCommand cmd;

// выделение памяти с инициализацией строки соединения с базой данных

cn = new SqlConnection(connectString);

cn.Open(); // открыть источник данных

cmd = cn.CreateCommand(); // задать SQL-команду

cmd.CommandText = SqlText; // задать командную строку

cmd.ExecuteNonQuery(); // выполнить SQL-команду

cn.Close(); // закрыть источник данных

}

}

}

NewClient.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class NewClient : Form

{

public NewClient()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

NewModel.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class NewModel : Form

{

public NewModel()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

NewOrder.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class NewOrder : Form

{

public NewOrder()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Не сохранять изменения?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Close();

/\* MainForm mainForm = new MainForm();

mainForm.ShowDialog();\*/

}

this.TopMost = true;

}

private void NewOrder\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Model". При необходимости она может быть перемещена или удалена.

this.modelTableAdapter.Fill(this.serviceStationDataSet.Model);

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Worker". При необходимости она может быть перемещена или удалена.

this.workerTableAdapter.Fill(this.serviceStationDataSet.Worker);

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Client". При необходимости она может быть перемещена или удалена.

this.clientTableAdapter.Fill(this.serviceStationDataSet.Client);

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Service". При необходимости она может быть перемещена или удалена.

this.serviceTableAdapter.Fill(this.serviceStationDataSet.Service);

}

private void button1\_Click(object sender, EventArgs e)

{

}

}

}

NewWorker.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class NewWorker : Form

{

public NewWorker()

{

InitializeComponent();

}

private void NewWorker\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Position". При необходимости она может быть перемещена или удалена.

this.positionTableAdapter.Fill(this.serviceStationDataSet.Position);

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Authorization());

}

}

}

UpdClient.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class UpdClient : Form

{

public UpdClient()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

UpdModel.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class UpdModel : Form

{

public UpdModel()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

UpdWorker.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProject

{

public partial class UpdWorker : Form

{

public UpdWorker()

{

InitializeComponent();

}

private void CloseBtn\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

}

}

Workers.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace CourseProject

{

public partial class Workers : Form

{

string connectString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\ServiceStation.mdf;Integrated Security=True";

public Workers()

{

InitializeComponent();

}

private void AllWorkers\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "serviceStationDataSet.Worker". При необходимости она может быть перемещена или удалена.

//this.workerTableAdapter.Fill(this.serviceStationDataSet.Worker);

SqlConnection myConnection = new SqlConnection(connectString);

myConnection.Open();

string query = "SELECT Worker.FIO, Worker.Age, Worker.Adress, Worker.PhoneNum, Worker.Passport, Position.PositionName FROM Worker INNER JOIN Position ON Worker.PK\_Position = Position.PositionID";

SqlCommand command = new SqlCommand(query, myConnection);

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

myConnection.Close();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

private void добавитб\_Click(object sender, EventArgs e)

{

string SqlText;

NewWorker newWorker = new NewWorker();

if (newWorker.ShowDialog() == DialogResult.OK)

{

//Regex regex = new Regex(@"^(A-Za-z)");

string regex = @"^(A-Za-z)";

Match match1 = Regex.Match(newWorker.textBox1.Text, regex);

Match match2 = Regex.Match(newWorker.textBox2.Text, regex);

Match match3 = Regex.Match(newWorker.textBox3.Text, regex);

Match match4 = Regex.Match(newWorker.textBox4.Text, regex);

Match match5 = Regex.Match(newWorker.textBox5.Text, regex);

// сформировать SQL-строку

if (match1.Success && match2.Success && match3.Success && match4.Success && match5.Success)

{

DialogResult result = MessageBox.Show(

"Вы ввели неверные данные. Повторите попытку.",

"Error",

MessageBoxButtons.YesNo,

MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

// выполнить SQL-команду

newWorker.Refresh();

}

this.TopMost = true;

}

else

{

SqlText = "INSERT INTO [Worker] ([FIO], [Age],[Adress],[PhoneNum],[Passport],[PK\_Position]) VALUES (";

SqlText = SqlText + "\'" + newWorker.textBox1.Text + "\', ";

SqlText = SqlText + "\'" + newWorker.textBox2.Text + "\', ";

SqlText = SqlText + "\'" + newWorker.textBox3.Text + "\', ";

SqlText = SqlText + "\'" + newWorker.textBox4.Text + "\', ";

SqlText = SqlText + "\'" + newWorker.textBox5.Text + "\', ";

SqlText = SqlText + "\'" + newWorker.comboBox1.ValueMember + "\')";

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

}

}

dataGridView1.Refresh();

}

private void редактировать\_Click(object sender, EventArgs e)

{

int index, n;

string SqlText = "UPDATE [Worker] SET ";

string ID\_Source, name, address;

// проверка, есть ли вообще записи в таблице Source

n = dataGridView1.Rows.Count;

if (n == 1) return;

UpdWorker f = new UpdWorker();

// заполнить форму данными перед открытием

index = dataGridView1.CurrentRow.Index;

ID\_Source = dataGridView1[0, index].Value.ToString();

name = dataGridView1[1, index].Value.ToString();

address = dataGridView1[2, index].Value.ToString();

}

private void удалить\_Click(object sender, EventArgs e)

{

int index, n;

string ID\_Client;

string SqlText = "DELETE FROM [Client] WHERE [Client].ClientID = ";

// проверка, есть ли вообще записи в таблице Source

n = dataGridView1.Rows.Count;

if (n == 1) return;

index = dataGridView1.CurrentRow.Index;

ID\_Client = Convert.ToString(dataGridView1[0, index].Value);

// сформировать SQL-команду

SqlText = SqlText + ID\_Client;

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

dataGridView1.Refresh();

}

this.TopMost = true;

}

private void выход\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Вы уверены?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

Close();

this.TopMost = true;

}

public void MyExecuteNonQuery(string SqlText)

{

SqlConnection cn; // экземпляр класса типа SqlConnection

SqlCommand cmd;

// выделение памяти с инициализацией строки соединения с базой данных

cn = new SqlConnection(connectString);

cn.Open(); // открыть источник данных

cmd = cn.CreateCommand(); // задать SQL-команду

cmd.CommandText = SqlText; // задать командную строку

cmd.ExecuteNonQuery(); // выполнить SQL-команду

cn.Close(); // закрыть источник данных

}

}

}

CreateTables.sql

CREATE TABLE dbo.Position

(

PositionID INT NOT NULL PRIMARY KEY,

PositionName VARCHAR(20) NULL,

Salary INT NULL,

Duties VARCHAR(100) NULL,

Requirements VARCHAR(100) NULL

);

CREATE TABLE dbo.Worker

(

WorkerID INT NOT NULL PRIMARY KEY,

FIO VARCHAR(100) NULL,

Age INT NULL,

Adress VARCHAR (50) NULL,

PhoneNum VARCHAR (15) NULL,

Passport VARCHAR(9) NULL,

PK\_Position INT NULL,

FOREIGN KEY (PK\_Position) REFERENCES Position(PositionID)

);

CREATE TABLE dbo.Model

(

ModelId INT NOT NULL PRIMARY KEY,

ModelName VARCHAR(50) NULL,

Manufacturer VARCHAR(20) NULL,

YearOfManufacture VARCHAR (50) NULL,

InspectionDate DATE NULL

);

CREATE TABLE dbo.ServicedShops

(

ServicedShopsId INT NOT NULL PRIMARY KEY,

ShopName VARCHAR(20) NULL,

Adress VARCHAR (50) NULL,

PhoneNum VARCHAR (11) NULL,

);

CREATE TABLE dbo.Service

(

ServiceId INT NOT NULL PRIMARY KEY,

ServiceName VARCHAR(20) NULL,

Price INT NULL,

);

CREATE TABLE dbo.Client

(

ClientID INT NOT NULL PRIMARY KEY,

FIO VARCHAR(100) NULL,

Adress VARCHAR (50) NULL,

PhoneNum VARCHAR (15) NULL

);

CREATE TABLE dbo.Spares

(

SparesId INT NOT NULL PRIMARY KEY,

SpareName VARCHAR(20) NULL,

Price INT NULL,

Existence BIT NULL,

PK\_Shop INT NULL,

FOREIGN KEY (PK\_Shop) REFERENCES ServicedShops(ServicedShopsId)

);

CREATE TABLE dbo.Orders

(

OrdersId INT NOT NULL PRIMARY KEY,

PK\_Service INT NULL,

PK\_Client INT NULL,

PK\_Worker INT NULL,

EtecutionMark VARCHAR (10) NULL,

Price INT NULL,

OrderDate DATE NULL,

ReturnDate DATE NULL,

PK\_Model INT NULL,

FOREIGN KEY (PK\_Service) REFERENCES Service(ServiceID),

FOREIGN KEY (PK\_Client) REFERENCES Client(ClientID),

FOREIGN KEY (PK\_Worker) REFERENCES Worker(WorkerID),

FOREIGN KEY (PK\_Model) REFERENCES Model(ModelID)

);

CREATE TABLE dbo.SpareManufacturer

(

SpareManufacturerId INT NOT NULL PRIMARY KEY,

ManufacturerName VARCHAR(20) NULL,

Adress VARCHAR(50) NULL,

PhoneNum VARCHAR(11) NULL,

PK\_Shop INT NULL,

FOREIGN KEY (PK\_Shop) REFERENCES ServicedShops(ServicedShopsId)

);

CREATE TABLE dbo.SpareForOrder

(

PK\_Orders INT NULL,

PK\_Spare INT NULL,

FOREIGN KEY (PK\_Spare) REFERENCES Spares(SparesId),

FOREIGN KEY (PK\_Orders) REFERENCES Orders(OrdersID)

);

CREATE TABLE dbo.Auth

(

Auth\_ID INT NOT NULL PRIMARY KEY,

AuthLogin VARCHAR(20) NULL,

AuthPass VARCHAR(20) NULL

);